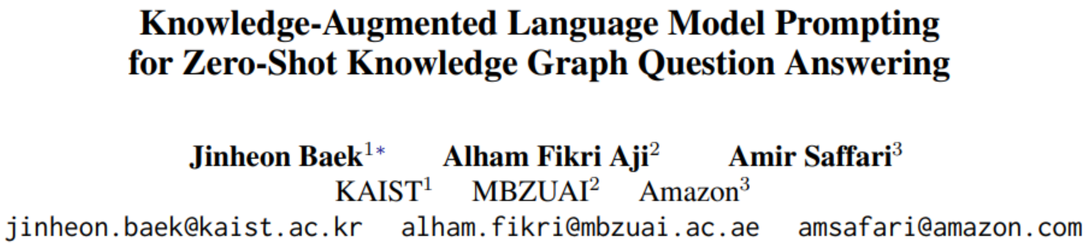
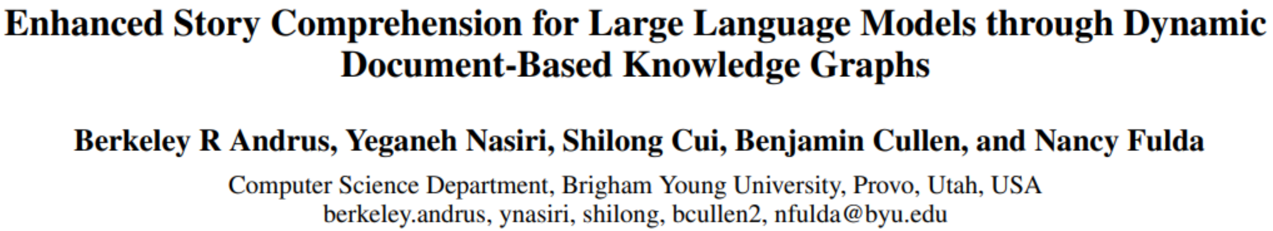


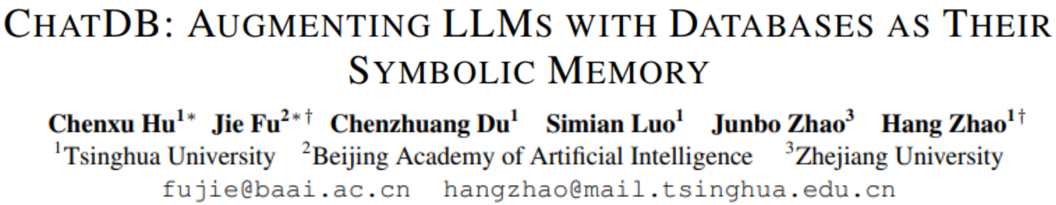
Knowledge-Augmented Language Model Prompting for Zero-Shot Knowledge Graph Question Answering

Jinheon Baek[1*] **Alham Fikri Aji**[2] **Amir Saffari**[3]
KAIST[1] MBZUAI[2] Amazon[3]
`jinheon.baek@kaist.ac.kr` `alham.fikri@mbzuai.ac.ae` `amsafari@amazon.com`



Enhanced Story Comprehension for Large Language Models through Dynamic Document-Based Knowledge Graphs

Berkeley R Andrus, Yeganeh Nasiri, Shilong Cui, Benjamin Cullen, and Nancy Fulda
Computer Science Department, Brigham Young University, Provo, Utah, USA
berkeley.andrus, ynasiri, shilong, bcullen2, nfulda@byu.edu



CHATDB: AUGMENTING LLMS WITH DATABASES AS THEIR SYMBOLIC MEMORY

Chenxu Hu[1*] **Jie Fu**[2*†] **Chenzhuang Du**[1] **Simian Luo**[1] **Junbo Zhao**[3] **Hang Zhao**[1†]
[1]Tsinghua University [2]Beijing Academy of Artificial Intelligence [3]Zhejiang University
`fujie@baai.ac.cn` `hangzhao@mail.tsinghua.edu.cn`

2023-08 여름 세미나
이재욱

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Knowledge-Augmented Language Model Prompting for Zero-Shot Knowledge Graph Question Answering

Jinheon Baek[1*] **Alham Fikri Aji**[2] **Amir Saffari**[3]
KAIST[1] MBZUAI[2] Amazon[3]
jinheon.baek@kaist.ac.kr alham.fikri@mbzuai.ac.ae amsafari@amazon.com

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Introduction

LLM은 pre-training 과정에서 parameter에 지식을 내재하고 있음
- 하지만 LLM이 내재하는 지식은 insufficient, incorrect, outdated
- 이것이 LLM이 잘못된 answer를 생성하는 원인이 됨 (Hallucination)

이를 해결하기 위해 주기적으로 LLM을 fine-tuning하여 지식을 update하는 것은 비용이 많이 소모

fine-tuning 없이, LLM의 input prompt에 relevant knowledge를 보강하면…?

Knowledge-Augmented Language Model Prompting(KAPING)
- model의 훈련 없이, zero-shot setting으로 KGQA task에서 성능을 검증

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Methodology

LM prompting for Zero-shot QA

- input question이 x가 주어지면, QA system은 output y를 반환해야 함
- x는 instruction template T를 거쳐 x'라는 prompt로 바뀜

여기서, naive zero-shot prompting의 문제점이 발생

1. LLM은 parameter에 학습된 지식에 의존하여 hallucination을 생성할 우려
2. Fine-tuning은 expensive
3. 어떤 지식을 통해 prompt에 대한 답을 생성하는지 알 수 없어 explainability가 떨어짐

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Methodology

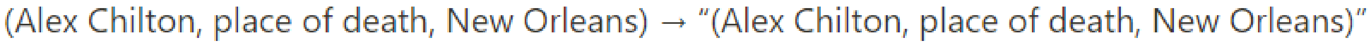
Knowledge-Augmented LM Prompting

LM prompting with KGs

- external KG에 접근하고 지식을 injection
- {subject, relation, object} 형태의 triples로 구성된 KG

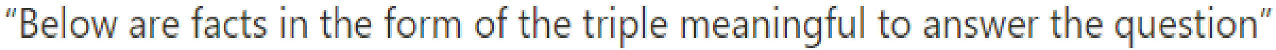
Knowledge Verbalization

- LLM은 textual input을 받으므로, {s, r, o} 형태의 triple을 그대로 이어붙이는 linear verbalization을 적용


(Alex Chilton, place of death, New Orleans) → “(Alex Chilton, place of death, New Orleans)”

Knowledge Injection

verbalized facts는 아래의 templat를 붙인 다음, input prompt x’ 에 concat하여 input으로 사용


“Below are facts in the form of the triple meaningful to answer the question”

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Methodology

Question-Relevant Knowledge Retrieval

verbalized KG는 text space에서 다룰 수 있다
-> embedding similarity를 계산할 수 있다!

off-the-shelf sentence embedding model(song et al., 2020, karpukhin et al., 2020; Xiong et al., 2021)로
임베딩한 다음 유사도를 계산

question-relevant knowledge는 question과 유사한 top-k(10) similar triples를 찾아 가져옴

off-the-shelf retriever를 불필요한 지식을 걸러내기 위한 도구로 사용
-> Limitations에 retriever 성능에 따라 generation 성능이 크게 좌우된다고 언급

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Experiments

평가에 사용한 데이터셋 (KGQA datasets)

: WebQuestionsSP(WebQSP), Mintaka

사용한 LLM
: T5(0.8, 3, 11B), T0(3, 11B), OPT(2.7, 6.7B), GPT-3(6.7, 175B)

제안하는 방법론(KAPING)과 비교하기 위해, 4가지 zero-shot LM prompting baseline을 설정

1. No Knowledge: w/o knowledge augmentation from KGs
2. Random Knowledge: question에 나타난 entity와 관련된 randomly sampled K triples를 prompt에 추가
3. Popular Knowledge: question entities의 모든 triples 중에서, 가장 자주 나타나는 relations를 기반으로 question entities의 모든 triples 중에서 K개의 popular triples를 input에 보강
4. Generated Knowledge: prompt를 기반으로 LLM 자체에서 지식을 생성한 후 input에 보강

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Experiments

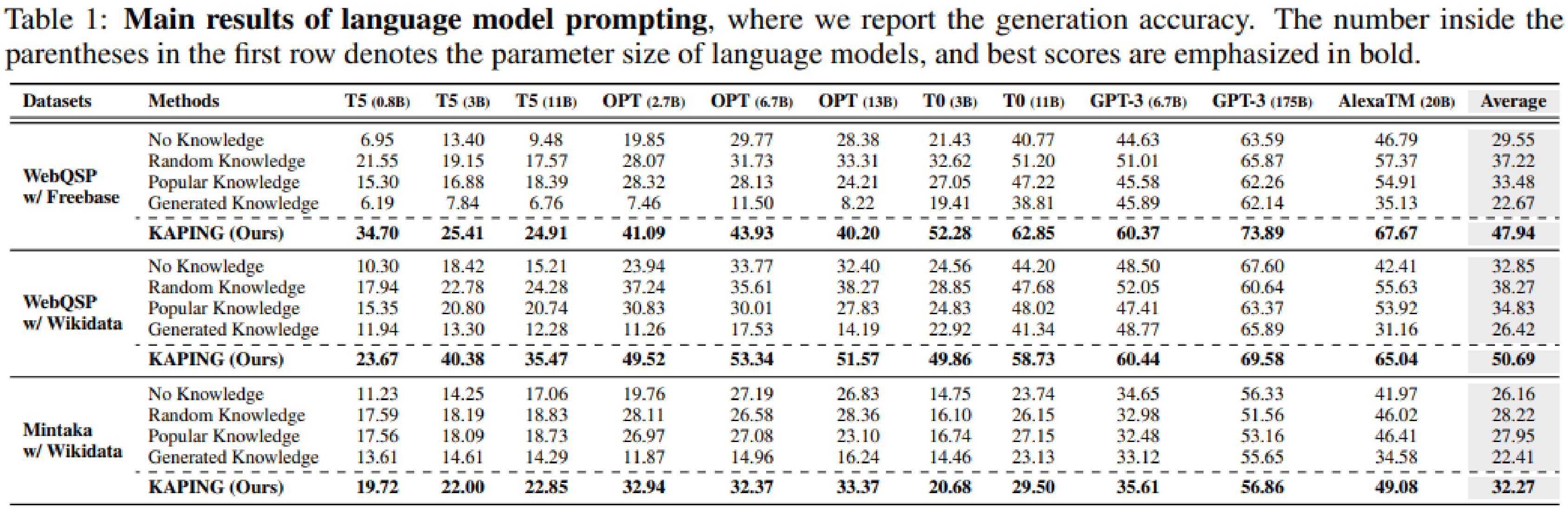


Table 1: **Main results of language model prompting**, where we report the generation accuracy. The number inside the parentheses in the first row denotes the parameter size of language models, and best scores are emphasized in bold.

Datasets	Methods	T5 (0.8B)	T5 (3B)	T5 (11B)	OPT (2.7B)	OPT (6.7B)	OPT (13B)	T0 (3B)	T0 (11B)	GPT-3 (6.7B)	GPT-3 (175B)	AlexaTM (20B)	Average
WebQSP w/ Freebase	No Knowledge	6.95	13.40	9.48	19.85	29.77	28.38	21.43	40.77	44.63	63.59	46.79	29.55
	Random Knowledge	21.55	19.15	17.57	28.07	31.73	33.31	32.62	51.20	51.01	65.87	57.37	37.22
	Popular Knowledge	15.30	16.88	18.39	28.32	28.13	24.21	27.05	47.22	45.58	62.26	54.91	33.48
	Generated Knowledge	6.19	7.84	6.76	7.46	11.50	8.22	19.41	38.81	45.89	62.14	35.13	22.67
	KAPING (Ours)	**34.70**	**25.41**	**24.91**	**41.09**	**43.93**	**40.20**	**52.28**	**62.85**	**60.37**	**73.89**	**67.67**	**47.94**
WebQSP w/ Wikidata	No Knowledge	10.30	18.42	15.21	23.94	33.77	32.40	24.56	44.20	48.50	67.60	42.41	32.85
	Random Knowledge	17.94	22.78	24.28	37.24	35.61	38.27	28.85	47.68	52.05	60.64	55.63	38.27
	Popular Knowledge	15.35	20.80	20.74	30.83	30.01	27.83	24.83	48.02	47.41	63.37	53.92	34.83
	Generated Knowledge	11.94	13.30	12.28	11.26	17.53	14.19	22.92	41.34	48.77	65.89	31.16	26.42
	KAPING (Ours)	**23.67**	**40.38**	**35.47**	**49.52**	**53.34**	**51.57**	**49.86**	**58.73**	**60.44**	**69.58**	**65.04**	**50.69**
Mintaka w/ Wikidata	No Knowledge	11.23	14.25	17.06	19.76	27.19	26.83	14.75	23.74	34.65	56.33	41.97	26.16
	Random Knowledge	17.59	18.19	18.83	28.11	26.58	28.36	16.10	26.15	32.98	51.56	46.02	28.22
	Popular Knowledge	17.56	18.09	18.73	26.97	27.08	23.10	16.74	27.15	32.48	53.16	46.41	27.95
	Generated Knowledge	13.61	14.61	14.29	11.87	14.96	16.24	14.46	23.13	33.12	55.65	34.58	22.41
	KAPING (Ours)	**19.72**	**22.00**	**22.85**	**32.94**	**32.37**	**33.37**	**20.68**	**29.50**	**35.61**	**56.86**	**49.08**	**32.27**

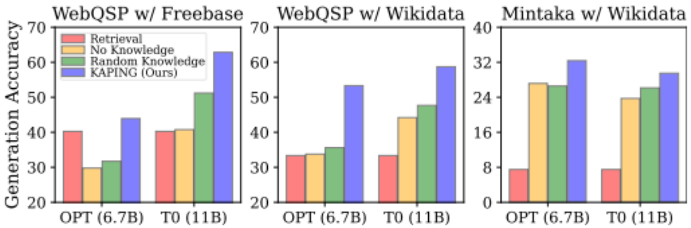


Figure 2: **Comparisons of retrieval and LM prompting.** Retrieval is the Top-1 result of the MPNet (Song et al., 2020).

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Experiments

생성 예시

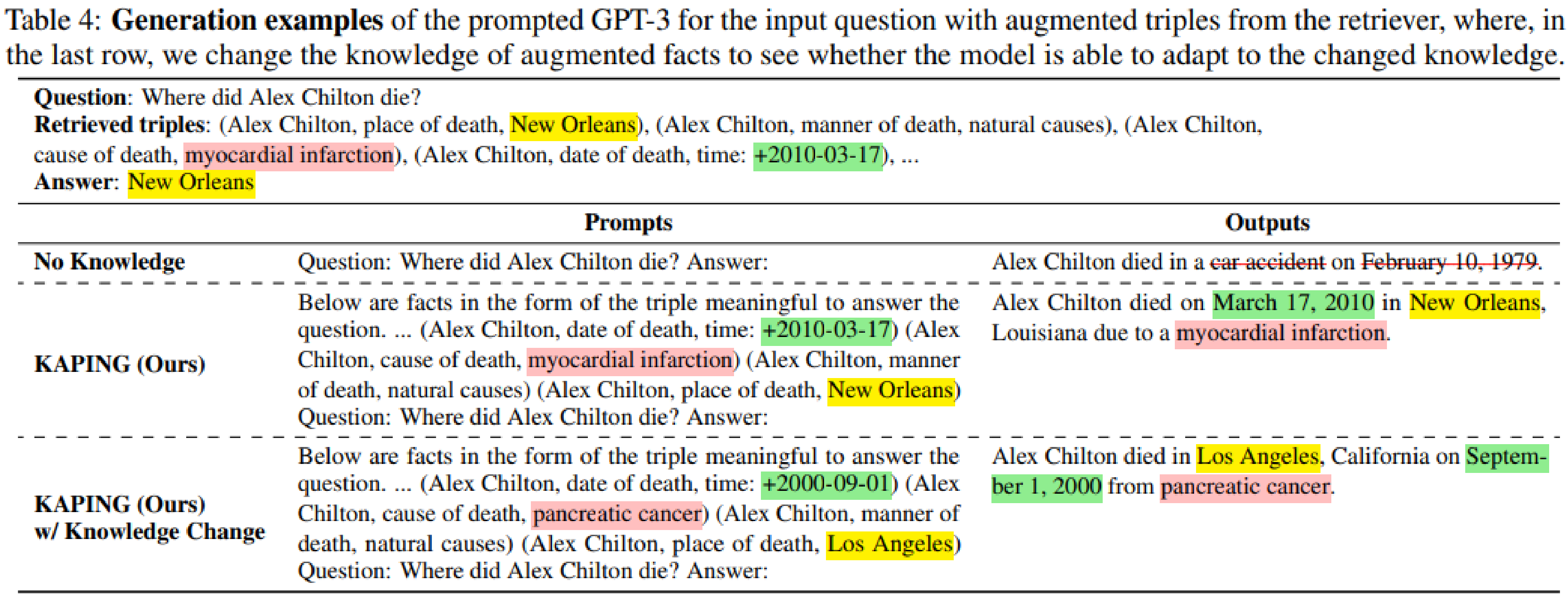


Table 4: **Generation examples** of the prompted GPT-3 for the input question with augmented triples from the retriever, where, in the last row, we change the knowledge of augmented facts to see whether the model is able to adapt to the changed knowledge.

Question: Where did Alex Chilton die?
Retrieved triples: (Alex Chilton, place of death, New Orleans), (Alex Chilton, manner of death, natural causes), (Alex Chilton, cause of death, myocardial infarction), (Alex Chilton, date of death, time: +2010-03-17), ...
Answer: New Orleans

	Prompts	**Outputs**
No Knowledge	Question: Where did Alex Chilton die? Answer:	Alex Chilton died in a ~~car accident~~ on ~~February 10, 1979~~.
KAPING (Ours)	Below are facts in the form of the triple meaningful to answer the question. ... (Alex Chilton, date of death, time: +2010-03-17) (Alex Chilton, cause of death, myocardial infarction) (Alex Chilton, manner of death, natural causes) (Alex Chilton, place of death, New Orleans) Question: Where did Alex Chilton die? Answer:	Alex Chilton died on March 17, 2010 in New Orleans, Louisiana due to a myocardial infarction.
KAPING (Ours) w/ Knowledge Change	Below are facts in the form of the triple meaningful to answer the question. ... (Alex Chilton, date of death, time: +2000-09-01) (Alex Chilton, cause of death, pancreatic cancer) (Alex Chilton, manner of death, natural causes) (Alex Chilton, place of death, Los Angeles) Question: Where did Alex Chilton die? Answer:	Alex Chilton died in Los Angeles, California on September 1, 2000 from pancreatic cancer.

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Enhanced Story Comprehension for Large Language Models through Dynamic Document-Based Knowledge Graphs

Berkeley R Andrus, Yeganeh Nasiri, Shilong Cui, Benjamin Cullen, and Nancy Fulda

Computer Science Department, Brigham Young University, Provo, Utah, USA
berkeley.andrus, ynasiri, shilong, bcullen2, nfulda@byu.edu

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Introduction

LLM은 story completion, comprehension, generation과 같은 다양한 story-related task에 사용됨

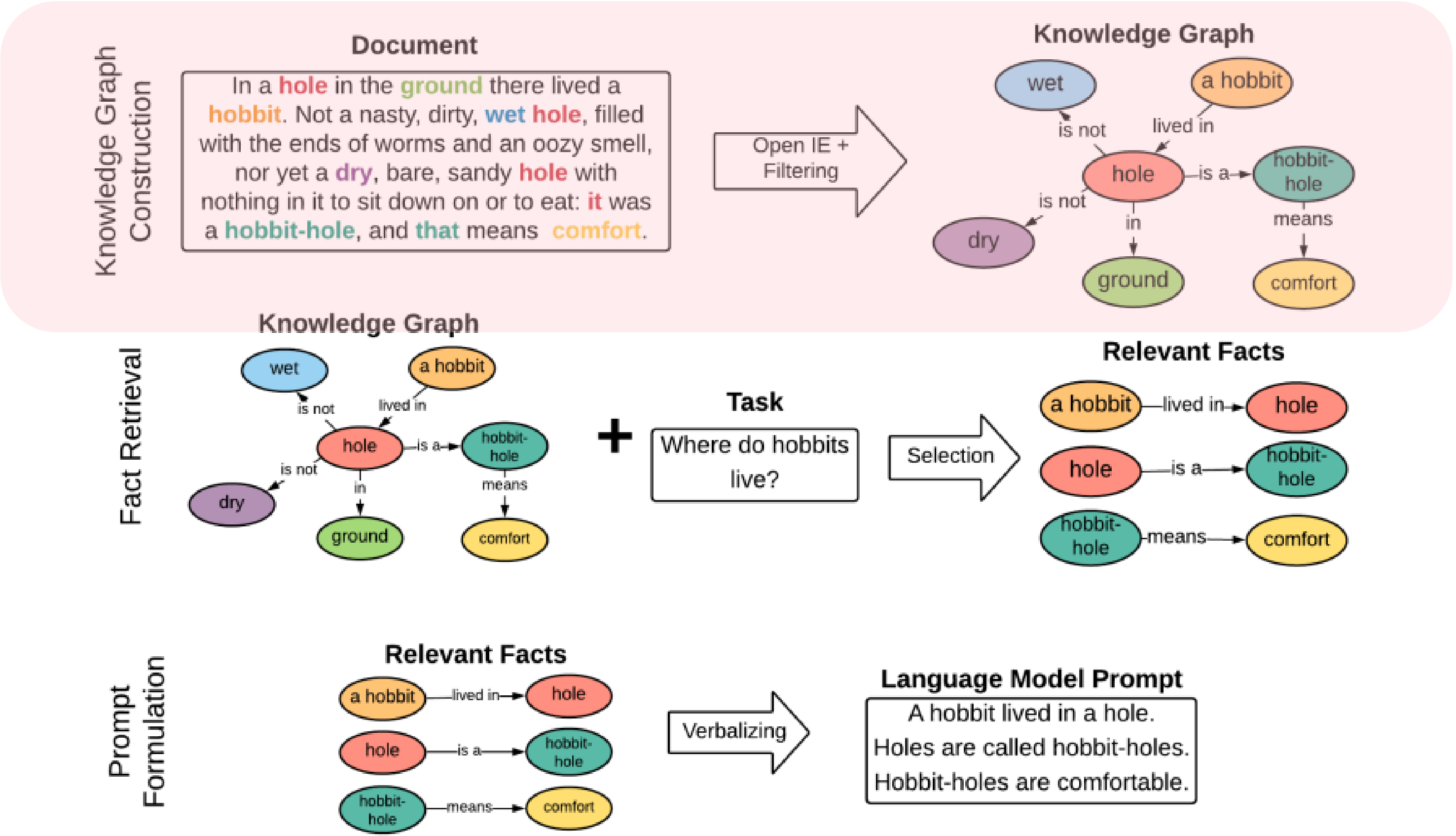
previous story-related research들은 model의 context window에 맞는 매우 짧은 길이의 story에 집중 (ex: ROCStories)

external KB를 통해 consistency를 높이려는 연구가 존재했지만, generated story 자체의 consistency에 집중한 것은 아님

본 연구에서는 documents에서 dynamic KB를 구축하고 이를 LLM의 Story Comprehension을 개선하기 위해 적용함

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Architecture



Document

In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it to sit down on or to eat: it was a hobbit-hole, and that means comfort.

Task

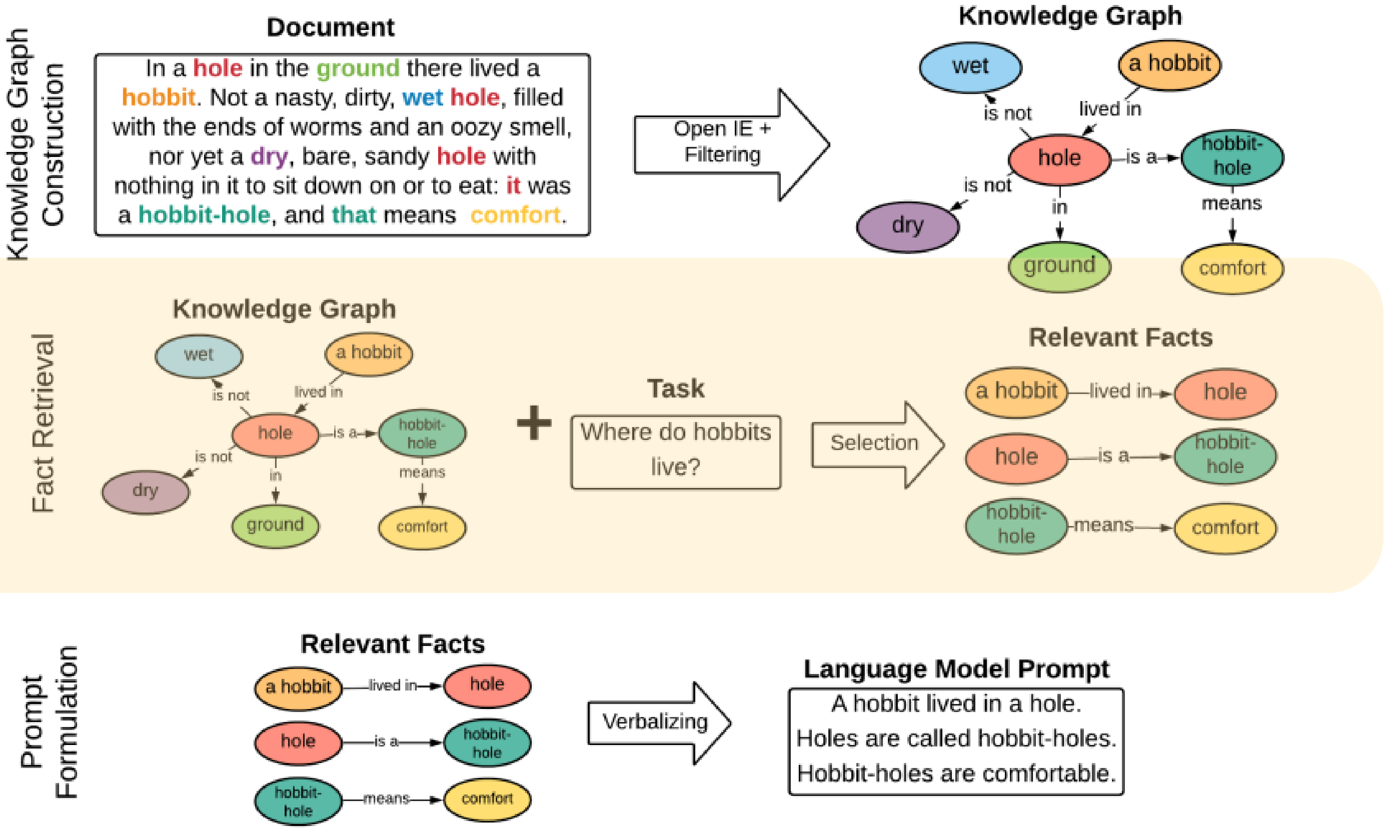
Where do hobbits live?

Language Model Prompt

A hobbit lived in a hole.
Holes are called hobbit-holes.
Hobbit-holes are comfortable.

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Architecture



Document

In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it to sit down on or to eat: it was a hobbit-hole, and that means comfort.

Task

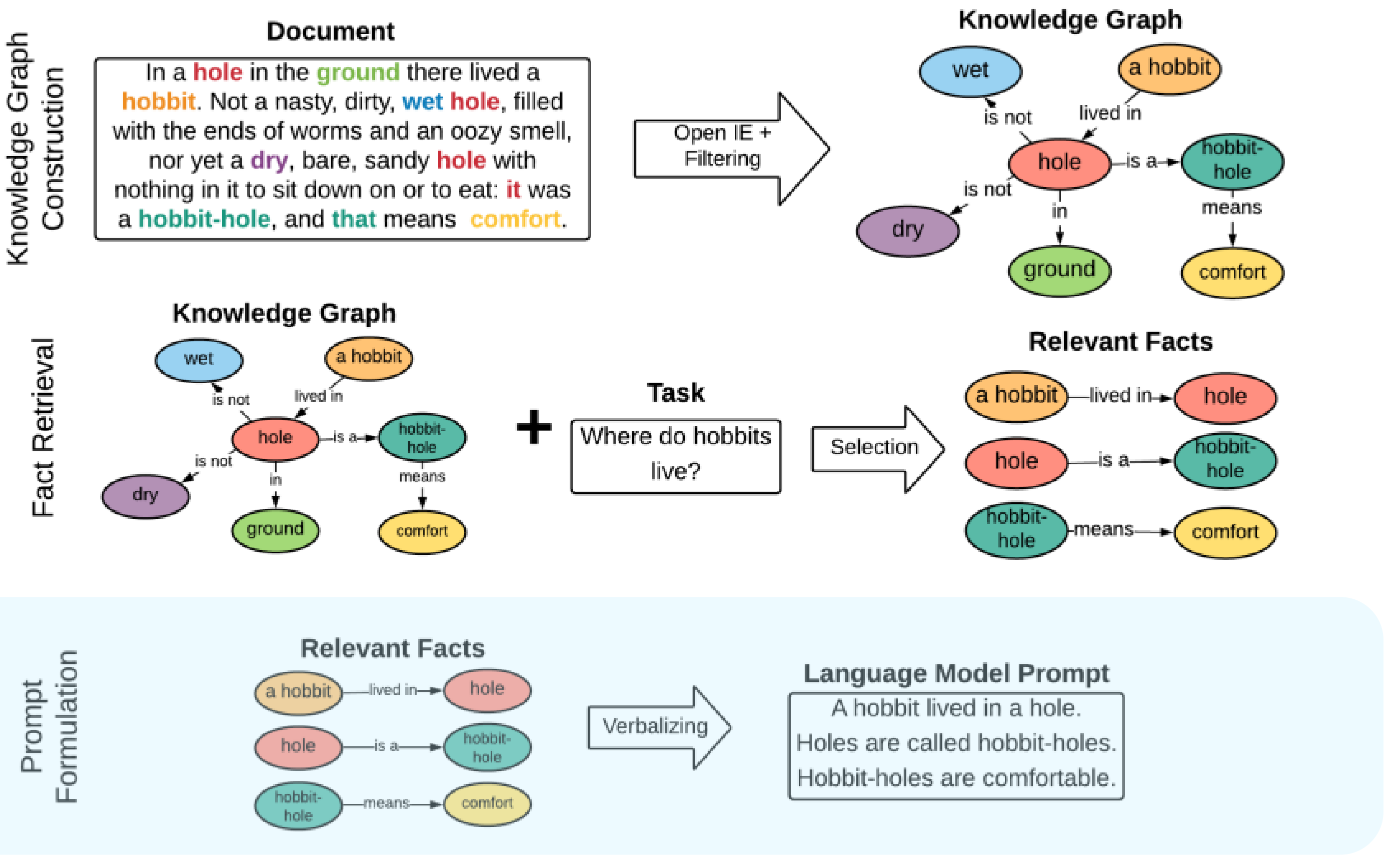
Where do hobbits live?

Language Model Prompt

A hobbit lived in a hole.
Holes are called hobbit-holes.
Hobbit-holes are comfortable.

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Architecture



Knowledge Graph Construction

Document

In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it to sit down on or to eat: it was a hobbit-hole, and that means comfort.

Open IE + Filtering

Knowledge Graph

Fact Retrieval

Knowledge Graph

+

Task

Where do hobbits live?

Selection

Relevant Facts

a hobbit → lived in → hole

hole → is a → hobbit-hole

hobbit-hole → means → comfort

Prompt Formulation

Relevant Facts

a hobbit → lived in → hole

hole → is a → hobbit-hole

hobbit-hole → means → comfort

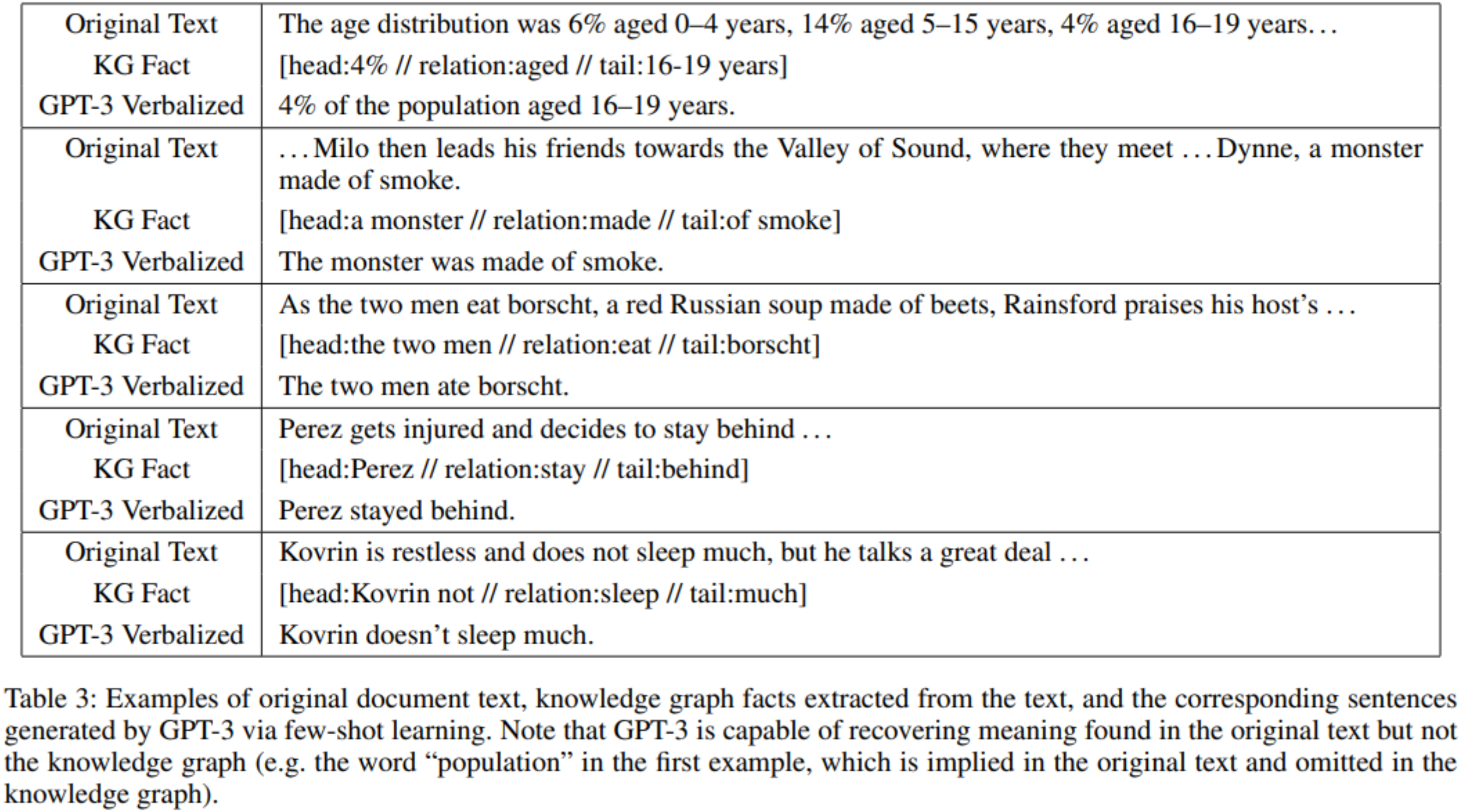
Verbalizing

Language Model Prompt

A hobbit lived in a hole.
Holes are called hobbit-holes.
Hobbit-holes are comfortable.

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

KG facts



Original Text	The age distribution was 6% aged 0–4 years, 14% aged 5–15 years, 4% aged 16–19 years…
KG Fact	[head:4% // relation:aged // tail:16-19 years]
GPT-3 Verbalized	4% of the population aged 16–19 years.
Original Text	…Milo then leads his friends towards the Valley of Sound, where they meet …Dynne, a monster made of smoke.
KG Fact	[head:a monster // relation:made // tail:of smoke]
GPT-3 Verbalized	The monster was made of smoke.
Original Text	As the two men eat borscht, a red Russian soup made of beets, Rainsford praises his host's …
KG Fact	[head:the two men // relation:eat // tail:borscht]
GPT-3 Verbalized	The two men ate borscht.
Original Text	Perez gets injured and decides to stay behind …
KG Fact	[head:Perez // relation:stay // tail:behind]
GPT-3 Verbalized	Perez stayed behind.
Original Text	Kovrin is restless and does not sleep much, but he talks a great deal …
KG Fact	[head:Kovrin not // relation:sleep // tail:much]
GPT-3 Verbalized	Kovrin doesn't sleep much.

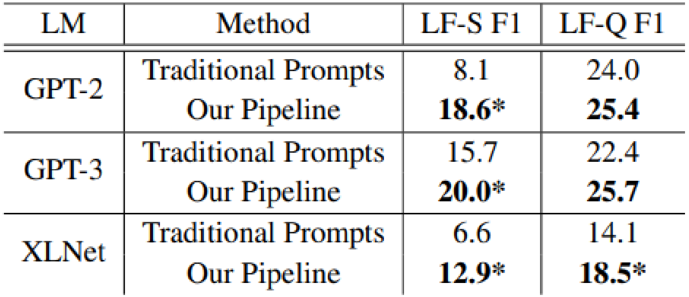
Table 3: Examples of original document text, knowledge graph facts extracted from the text, and the corresponding sentences generated by GPT-3 via few-shot learning. Note that GPT-3 is capable of recovering meaning found in the original text but not the knowledge graph (e.g. the word "population" in the first example, which is implied in the original text and omitted in the knowledge graph).

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Experiments

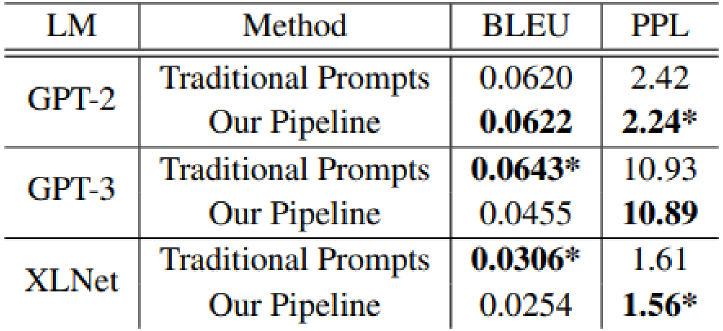
실험으로는 GPT-2, GPT-3, XLNet을 사용하여 QA와 Story Completion을 수행

LF-SQuAD, LF-QUOREF라는 새로운 데이터셋으로 평가
- 스토리 당 1166.3개 토큰 (ROCStories: 53.5 avg tokens)



LM	Method	LF-S F1	LF-Q F1
GPT-2	Traditional Prompts	8.1	24.0
	Our Pipeline	**18.6***	**25.4**
GPT-3	Traditional Prompts	15.7	22.4
	Our Pipeline	**20.0***	**25.7**
XLNet	Traditional Prompts	6.6	14.1
	Our Pipeline	**12.9***	**18.5***

Table 1: Results of question answering experiments described in Section 4.1. LF-S and LF-Q are LF-SQuAD and LF-QUOREF respectively. F1 scores in this context measure the precision and recall between bag-of-words tokens used in a human-written and predicted answer. Bolded scores are better and * indicates a significant result ($\alpha = 0.01$).



LM	Method	BLEU	PPL
GPT-2	Traditional Prompts	0.0620	2.42
	Our Pipeline	**0.0622**	**2.24***
GPT-3	Traditional Prompts	**0.0643***	10.93
	Our Pipeline	0.0455	**10.89**
XLNet	Traditional Prompts	**0.0306***	1.61
	Our Pipeline	0.0254	**1.56***

Table 2: Results of story completion experiments on the Sparknotes dataset described in Section 4.2. Bolded scores are better (higher for BLEU, lower for PPL) and * indicates a significant result ($\alpha = 0.01$).

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

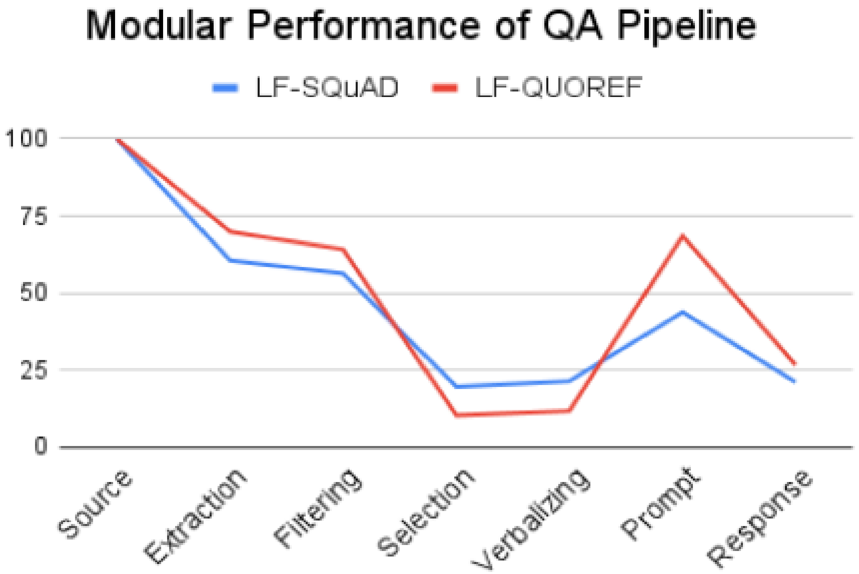
Module Analysis

정보 손실이 가장 많이 일어나는 구간
- Selection
- Response

-> 해당 단계에서 가장 많은 양의 정보가 걸러지기 때문

필터링 단계에서는 정보 손실이 거의 X
- 필요 없는 facts를 거르는 데 도움

Verbalizing 모듈은 시스템이 정답을 유지하는 사례의 비율이 높은 것으로 나타남
- selected facts를 GPT-3가 언어화 하는 과정에서 LLM이 지식을 해석하는데 중요한 기능을 하는 기능어나 서식을 추가하는 경우 때문이라고 해석



Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

CHATDB: AUGMENTING LLMS WITH DATABASES AS THEIR SYMBOLIC MEMORY

Chenxu Hu[1]* **Jie Fu**[2]*† **Chenzhuang Du**[1] **Simian Luo**[1] **Junbo Zhao**[3] **Hang Zhao**[1]†
[1]Tsinghua University [2]Beijing Academy of Artificial Intelligence [3]Zhejiang University
`fujie@baai.ac.cn` `hangzhao@mail.tsinghua.edu.cn`

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Introduction

GPT-4, PALM2와 같은 LLM은 문맥을 이해하고 관련된 응답을 생성하는 능력이 매우 강력함
하지만, 사용자와의 상호작용이 길어질수록, 모델 자체의 input token limitation를 초과하는 한계가 있음

상호작용이 길어져 context 정보를 단순히 연결하여 LLM에 밀어 넣으면?
-> LLM의 처리 능력을 초과
-> 오류가 누적
-> 정확도가 떨어지는 응답을 생성

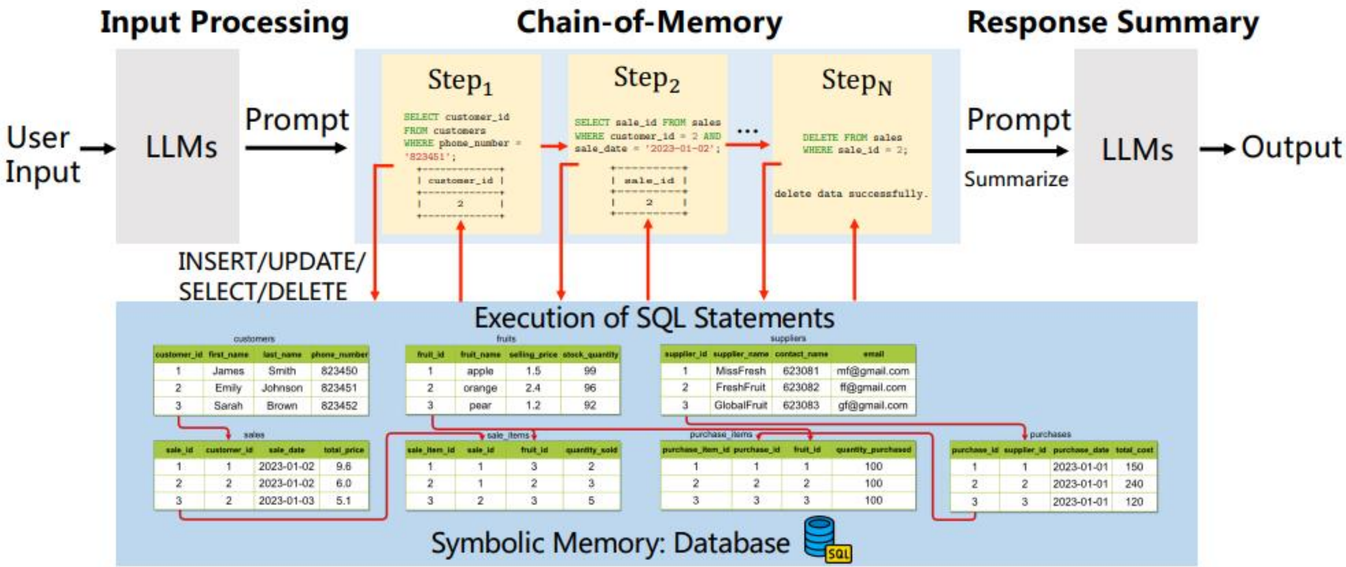
limited input length 문제를 해결하기 위해, 이전 상호작용의 관련 정보를 저장하고 검색하는 neural memory mechanism이 고안되었으나, 아래와 같은 이유로 특히 multi-hop 추론에서 history 검색에 어려움을 겪음

- history의 정보를 구조화된 형태로 저장하지 않음
- memory의 정보를 조작할 때, 부정확할 수 있는 Vector similarity calculation에 의존

본 논문에서는 DB를 symbolic memory로 활용하는 framework인 ChatDB를 제안
- Chain-of-Memory(CoM)을 적용하여 복잡한 multi-hop 추론을 여러 단계의 메모리 연산으로 분해

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

ChatDB



Step$_1$

```
SELECT customer_id
FROM customers
WHERE phone_number =
'823451';
```

```
+-------------+
| customer_id |
+-------------+
|      2      |
+-------------+
```

Step$_2$

```
SELECT sale_id FROM sales
WHERE customer_id = 2 AND
sale_date = '2023-01-02';
```

```
+---------+
| sale_id |
+---------+
|    2    |
+---------+
```

Step$_N$

```
DELETE FROM sales
WHERE sale_id = 2;
```

```
delete data successfully.
```

customers

customer_id	first_name	last_name	phone_number
1	James	Smith	823450
2	Emily	Johnson	823451
3	Sarah	Brown	823452

fruits

fruit_id	fruit_name	selling_price	stock_quantity
1	apple	1.5	99
2	orange	2.4	96
3	pear	1.2	92

suppliers

supplier_id	supplier_name	contact_name	email
1	MissFresh	623081	mf@gmail.com
2	FreshFruit	623082	ff@gmail.com
3	GlobalFruit	623083	gf@gmail.com

sales

sale_id	customer_id	sale_date	total_price
1	1	2023-01-02	9.6
2	2	2023-01-02	6.0
3	2	2023-01-03	5.1

sale_items

sale_item_id	sale_id	fruit_id	quantity_sold
1	1	3	2
2	1	2	3
3	2	3	5

purchase_items

purchase_item_id	purchase_id	fruit_id	quantity_purchased
1	1	1	100
2	2	2	100
3	3	3	100

purchases

purchase_id	supplier_id	purchase_date	total_cost
1	1	2023-01-01	150
2	2	2023-01-01	240
3	3	2023-01-01	120

Figure 2: ChatDB framework. The red arrow lines represent the process flow of chain-of-memory, indicating the connection between multiple memory operations. The red arrow lines between database tables represent the reference relationships between primary keys and foreign keys, which start from primary keys to foreign keys. Only the first four columns of each table are shown for brevity. This example showcases the process of returning goods purchased on 2023-01-02 by a customer with the phone number 823451.

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Comparison

with previous memory-augmented LLM

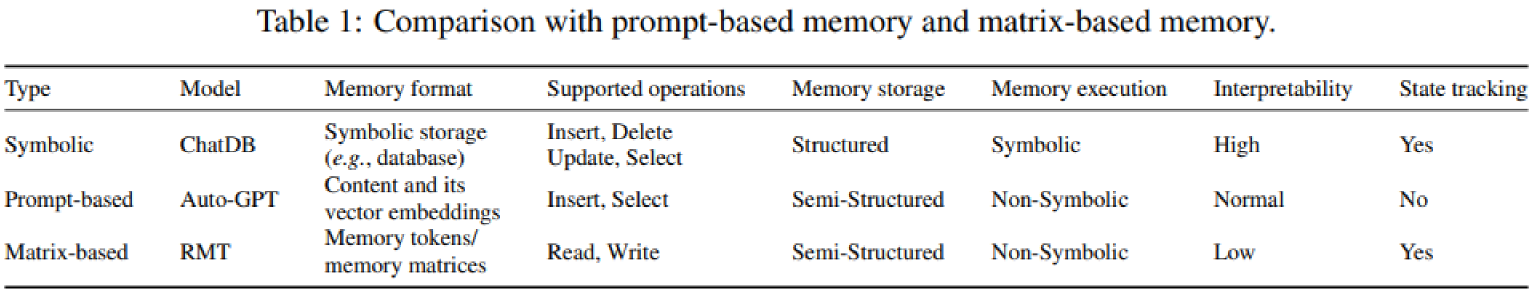


Table 1: Comparison with prompt-based memory and matrix-based memory.

Type	Model	Memory format	Supported operations	Memory storage	Memory execution	Interpretability	State tracking
Symbolic	ChatDB	Symbolic storage (*e.g.*, database)	Insert, Delete Update, Select	Structured	Symbolic	High	Yes
Prompt-based	Auto-GPT	Content and its vector embeddings	Insert, Select	Semi-Structured	Non-Symbolic	Normal	No
Matrix-based	RMT	Memory tokens/ memory matrices	Read, Write	Semi-Structured	Non-Symbolic	Low	Yes

Prompt-based memory:

- context를 저장하고, retrieval 모델을 사용하여 현재 대화와 가장 연관된 past interaction의 content를 찾아 이를 언어 모델의 prompt로 사용

Matrix-based memory:

- additional memory tokens or memory matrices를 메모리로 활용

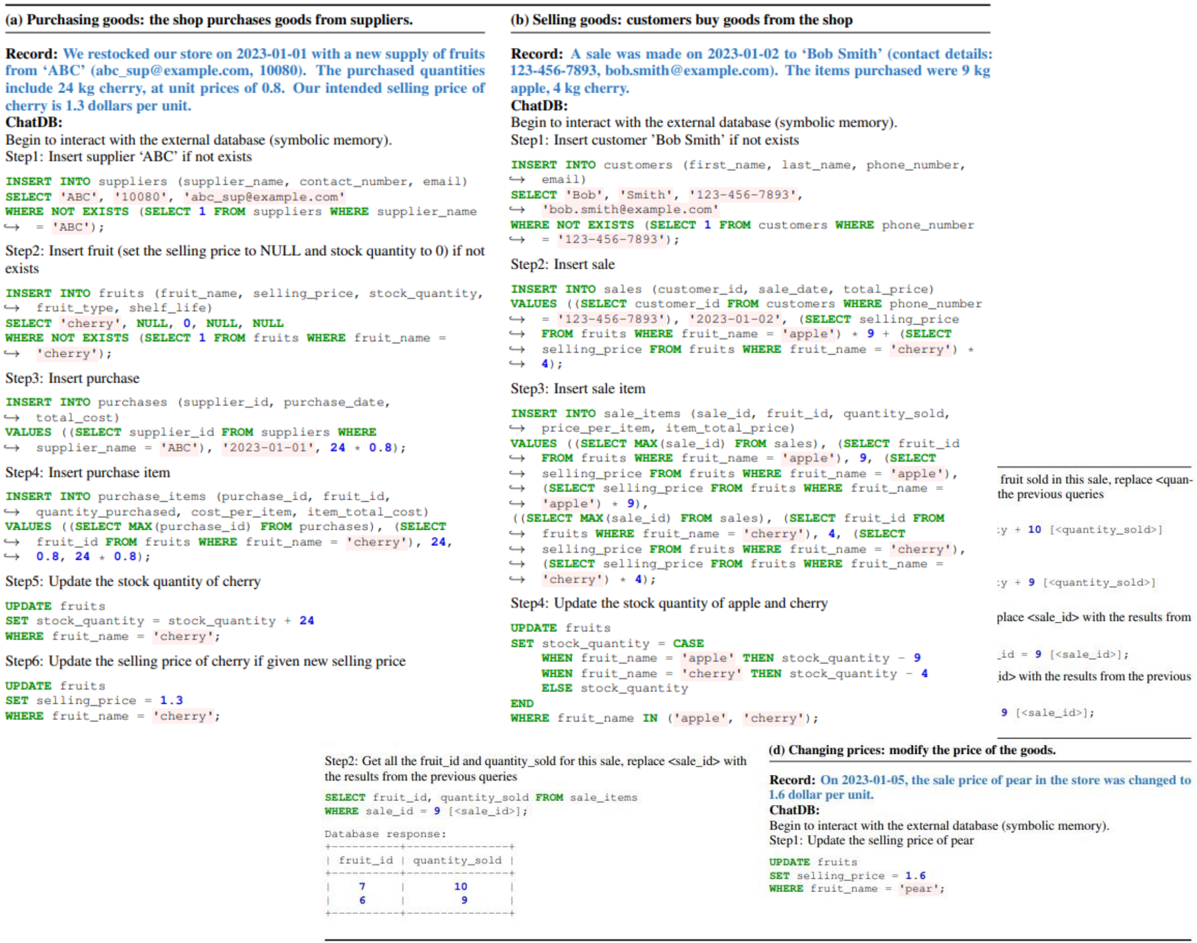
Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Evaluation

과거 정보를 정확히 기록하고 처리하는 시나리오
- 과일 가게의 경영을 시뮬레이션하는 합성 데이터세트

모델 성능 평가를 위한 50개의 질문 세트
- 과거 데이터에서 정보를 검색하는 쉬운 문제부터 multi-hop 추론이 필요한 어려운 문제까지 다양한 난이도로 구성

ChatDB vs. ChatGPT



(a) Purchasing goods: the shop purchases goods from suppliers.

Record: We restocked our store on 2023-01-01 with a new supply of fruits from 'ABC' (abc_sup@example.com, 10080). The purchased quantities include 24 kg cherry, at unit prices of 0.8. Our intended selling price of cherry is 1.3 dollars per unit.
ChatDB:
Begin to interact with the external database (symbolic memory).
Step1: Insert supplier 'ABC' if not exists

```
INSERT INTO suppliers (supplier_name, contact_number, email)
SELECT 'ABC', '10080', 'abc_sup@example.com'
WHERE NOT EXISTS (SELECT 1 FROM suppliers WHERE supplier_name
↪  = 'ABC');
```

Step2: Insert fruit (set the selling price to NULL and stock quantity to 0) if not exists

```
INSERT INTO fruits (fruit_name, selling_price, stock_quantity,
↪  fruit_type, shelf_life)
SELECT 'cherry', NULL, 0, NULL, NULL
WHERE NOT EXISTS (SELECT 1 FROM fruits WHERE fruit_name =
↪  'cherry');
```

Step3: Insert purchase

```
INSERT INTO purchases (supplier_id, purchase_date,
↪  total_cost)
VALUES ((SELECT supplier_id FROM suppliers WHERE
↪  supplier_name = 'ABC'), '2023-01-01', 24 * 0.8);
```

Step4: Insert purchase item

```
INSERT INTO purchase_items (purchase_id, fruit_id,
↪  quantity_purchased, cost_per_item, item_total_cost)
VALUES ((SELECT MAX(purchase_id) FROM purchases), (SELECT
↪  fruit_id FROM fruits WHERE fruit_name = 'cherry'), 24,
↪  0.8, 24 * 0.8);
```

Step5: Update the stock quantity of cherry

```
UPDATE fruits
SET stock_quantity = stock_quantity + 24
WHERE fruit_name = 'cherry';
```

Step6: Update the selling price of cherry if given new selling price

```
UPDATE fruits
SET selling_price = 1.3
WHERE fruit_name = 'cherry';
```

(b) Selling goods: customers buy goods from the shop

Record: A sale was made on 2023-01-02 to 'Bob Smith' (contact details: 123-456-7893, bob.smith@example.com). The items purchased were 9 kg apple, 4 kg cherry.
ChatDB:
Begin to interact with the external database (symbolic memory).
Step1: Insert customer 'Bob Smith' if not exists

```
INSERT INTO customers (first_name, last_name, phone_number,
↪  email)
SELECT 'Bob', 'Smith', '123-456-7893',
↪  'bob.smith@example.com'
WHERE NOT EXISTS (SELECT 1 FROM customers WHERE phone_number
↪  = '123-456-7893');
```

Step2: Insert sale

```
INSERT INTO sales (customer_id, sale_date, total_price)
VALUES ((SELECT customer_id FROM customers WHERE phone_number
↪  = '123-456-7893'), '2023-01-02', (SELECT selling_price
↪  FROM fruits WHERE fruit_name = 'apple') * 9 + (SELECT
↪  selling_price FROM fruits WHERE fruit_name = 'cherry') *
↪  4);
```

Step3: Insert sale item

```
INSERT INTO sale_items (sale_id, fruit_id, quantity_sold,
↪  price_per_item, item_total_price)
VALUES ((SELECT MAX(sale_id) FROM sales), (SELECT fruit_id
↪  FROM fruits WHERE fruit_name = 'apple'), 9, (SELECT
↪  selling_price FROM fruits WHERE fruit_name = 'apple'),
↪  (SELECT selling_price FROM fruits WHERE fruit_name =
↪  'apple') * 9),
((SELECT MAX(sale_id) FROM sales), (SELECT fruit_id FROM
↪  fruits WHERE fruit_name = 'cherry'), 4, (SELECT
↪  selling_price FROM fruits WHERE fruit_name = 'cherry'),
↪  (SELECT selling_price FROM fruits WHERE fruit_name =
↪  'cherry') * 4);
```

Step4: Update the stock quantity of apple and cherry

```
UPDATE fruits
SET stock_quantity = CASE
    WHEN fruit_name = 'apple' THEN stock_quantity - 9
    WHEN fruit_name = 'cherry' THEN stock_quantity - 4
    ELSE stock_quantity
END
WHERE fruit_name IN ('apple', 'cherry');
```

fruit sold in this sale, replace <quan-
the previous queries

```
y + 10 [<quantity_sold>]
```

```
y + 9 [<quantity_sold>]
```

place <sale_id> with the results from

```
id = 9 [<sale_id>];
```

id> with the results from the previous

```
9 [<sale_id>];
```

Step2: Get all the fruit_id and quantity_sold for this sale, replace <sale_id> with the results from the previous queries

```
SELECT fruit_id, quantity_sold FROM sale_items
WHERE sale_id = 9 [<sale_id>];
```

```
Database response:
+----------+--------------+
| fruit_id | quantity_sold |
+----------+--------------+
|    7     |      10      |
|    6     |       9      |
+----------+--------------+
```

(d) Changing prices: modify the price of the goods.

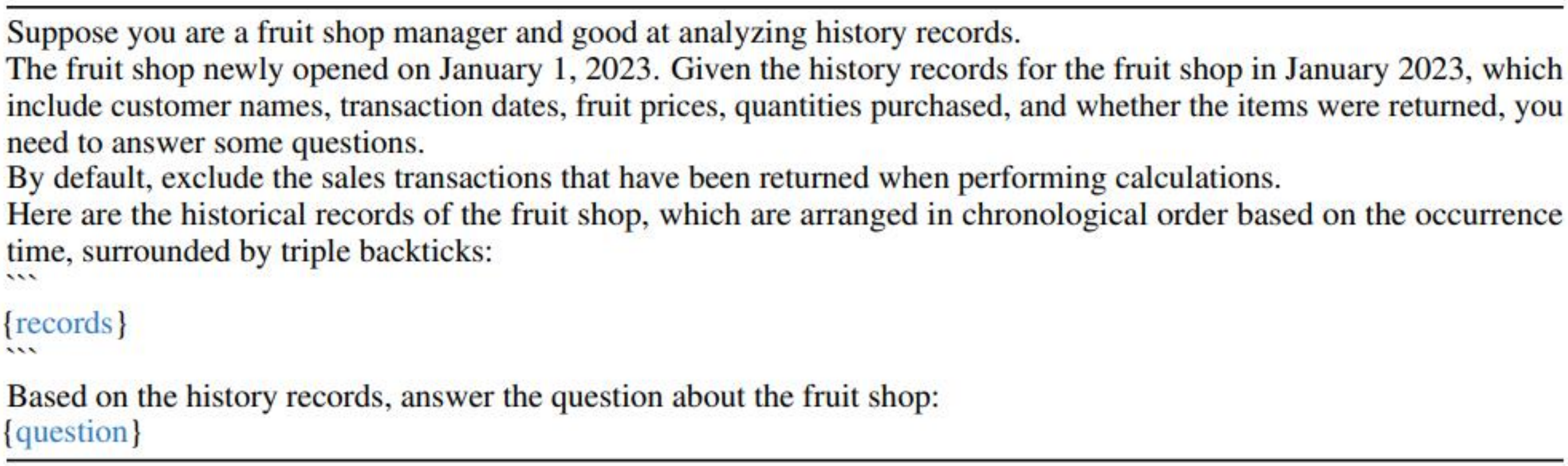
Record: On 2023-01-05, the sale price of pear in the store was changed to 1.6 dollar per unit.
ChatDB:
Begin to interact with the external database (symbolic memory).
Step1: Update the selling price of pear

```
UPDATE fruits
SET selling_price = 1.6
WHERE fruit_name = 'pear';
```

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Evaluation

- ChatGPT prompt

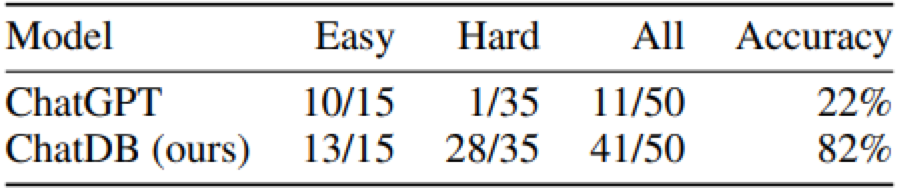


~~~
Suppose you are a fruit shop manager and good at analyzing history records.
The fruit shop newly opened on January 1, 2023. Given the history records for the fruit shop in January 2023, which include customer names, transaction dates, fruit prices, quantities purchased, and whether the items were returned, you need to answer some questions.
By default, exclude the sales transactions that have been returned when performing calculations.
Here are the historical records of the fruit shop, which are arranged in chronological order based on the occurrence time, surrounded by triple backticks:
```
{records}
```
Based on the history records, answer the question about the fruit shop:
{question}
~~~

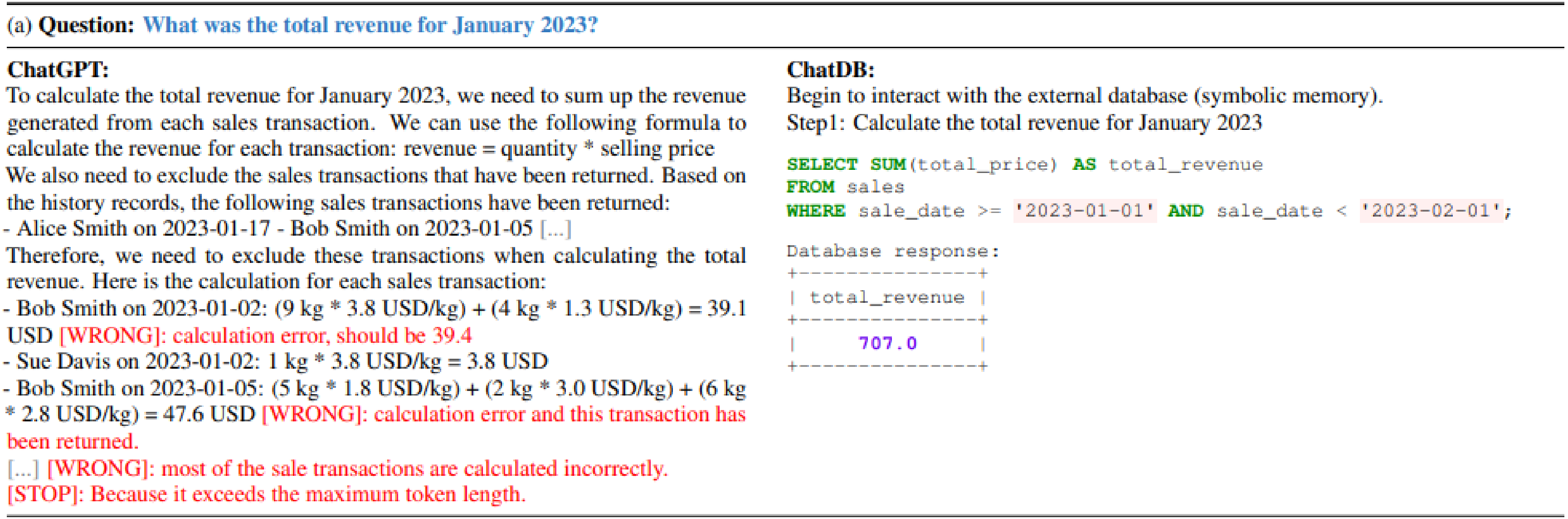
Figure 4: Prompt for ChatGPT to answer questions in the Fruit Shop Dataset. The placeholders “records” and “question” will be replaced by specific details when it is actually used.

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Evaluation



Model	Easy	Hard	All	Accuracy
ChatGPT	10/15	1/35	11/50	22%
ChatDB (ours)	13/15	28/35	41/50	82%



(a) **Question:** What was the total revenue for January 2023?

ChatGPT:
To calculate the total revenue for January 2023, we need to sum up the revenue generated from each sales transaction. We can use the following formula to calculate the revenue for each transaction: revenue = quantity * selling price
We also need to exclude the sales transactions that have been returned. Based on the history records, the following sales transactions have been returned:
- Alice Smith on 2023-01-17 - Bob Smith on 2023-01-05 [...]
Therefore, we need to exclude these transactions when calculating the total revenue. Here is the calculation for each sales transaction:
- Bob Smith on 2023-01-02: (9 kg * 3.8 USD/kg) + (4 kg * 1.3 USD/kg) = 39.1 USD [WRONG]: calculation error, should be 39.4
- Sue Davis on 2023-01-02: 1 kg * 3.8 USD/kg = 3.8 USD
- Bob Smith on 2023-01-05: (5 kg * 1.8 USD/kg) + (2 kg * 3.0 USD/kg) + (6 kg * 2.8 USD/kg) = 47.6 USD [WRONG]: calculation error and this transaction has been returned.
[...] [WRONG]: most of the sale transactions are calculated incorrectly.
[STOP]: Because it exceeds the maximum token length.

ChatDB:
Begin to interact with the external database (symbolic memory).
Step1: Calculate the total revenue for January 2023

```
SELECT SUM(total_price) AS total_revenue
FROM sales
WHERE sale_date >= '2023-01-01' AND sale_date < '2023-02-01';
```

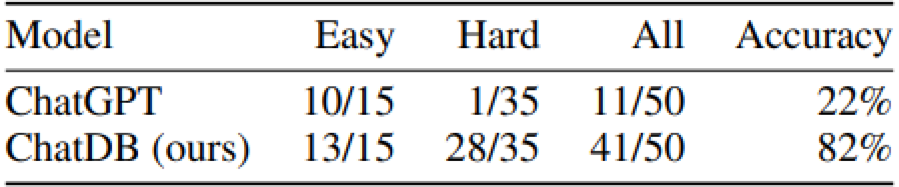
```
Database response:
+----------------+
| total_revenue |
+----------------+
|     707.0      |
+----------------+
```

ChatDB가 훨씬 더 높은 정확도로 ChatGPT보다 성능이 뛰어나다는 것을 보임

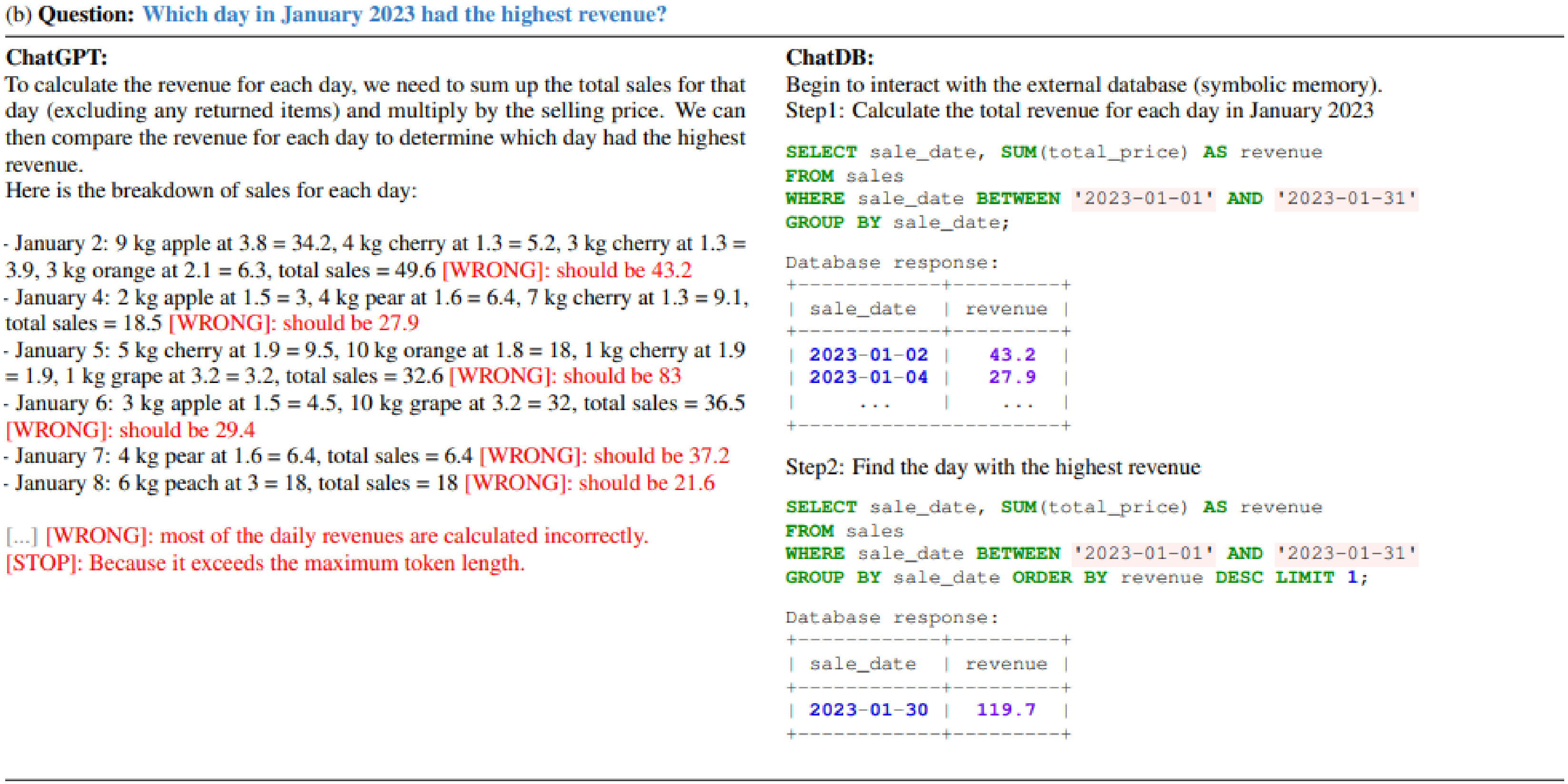
ChatGPT는 쉬운 문제는 잘 푸나, multi-hop reasoning이나 정확한 계산이 필요한 어려운 질문은 잘 못 푸는 경향

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Evaluation



Model	Easy	Hard	All	Accuracy
ChatGPT	10/15	1/35	11/50	22%
ChatDB (ours)	13/15	28/35	41/50	82%



(b) **Question:** Which day in January 2023 had the highest revenue?

ChatGPT:

To calculate the revenue for each day, we need to sum up the total sales for that day (excluding any returned items) and multiply by the selling price. We can then compare the revenue for each day to determine which day had the highest revenue.

Here is the breakdown of sales for each day:

- January 2: 9 kg apple at 3.8 = 34.2, 4 kg cherry at 1.3 = 5.2, 3 kg cherry at 1.3 = 3.9, 3 kg orange at 2.1 = 6.3, total sales = 49.6 [WRONG]: should be 43.2
- January 4: 2 kg apple at 1.5 = 3, 4 kg pear at 1.6 = 6.4, 7 kg cherry at 1.3 = 9.1, total sales = 18.5 [WRONG]: should be 27.9
- January 5: 5 kg cherry at 1.9 = 9.5, 10 kg orange at 1.8 = 18, 1 kg cherry at 1.9 = 1.9, 1 kg grape at 3.2 = 3.2, total sales = 32.6 [WRONG]: should be 83
- January 6: 3 kg apple at 1.5 = 4.5, 10 kg grape at 3.2 = 32, total sales = 36.5 [WRONG]: should be 29.4
- January 7: 4 kg pear at 1.6 = 6.4, total sales = 6.4 [WRONG]: should be 37.2
- January 8: 6 kg peach at 3 = 18, total sales = 18 [WRONG]: should be 21.6

[...] [WRONG]: most of the daily revenues are calculated incorrectly.
[STOP]: Because it exceeds the maximum token length.

ChatDB:

Begin to interact with the external database (symbolic memory).
Step1: Calculate the total revenue for each day in January 2023

```
SELECT sale_date, SUM(total_price) AS revenue
FROM sales
WHERE sale_date BETWEEN '2023-01-01' AND '2023-01-31'
GROUP BY sale_date;
```

```
Database response:
+------------+---------+
| sale_date  | revenue |
+------------+---------+
| 2023-01-02 |   43.2  |
| 2023-01-04 |   27.9  |
|    ...     |   ...   |
+----------------------+
```

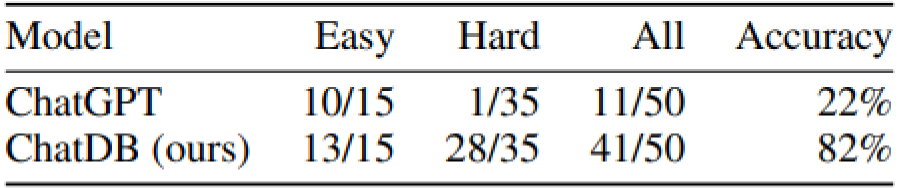
Step2: Find the day with the highest revenue

```
SELECT sale_date, SUM(total_price) AS revenue
FROM sales
WHERE sale_date BETWEEN '2023-01-01' AND '2023-01-31'
GROUP BY sale_date ORDER BY revenue DESC LIMIT 1;
```

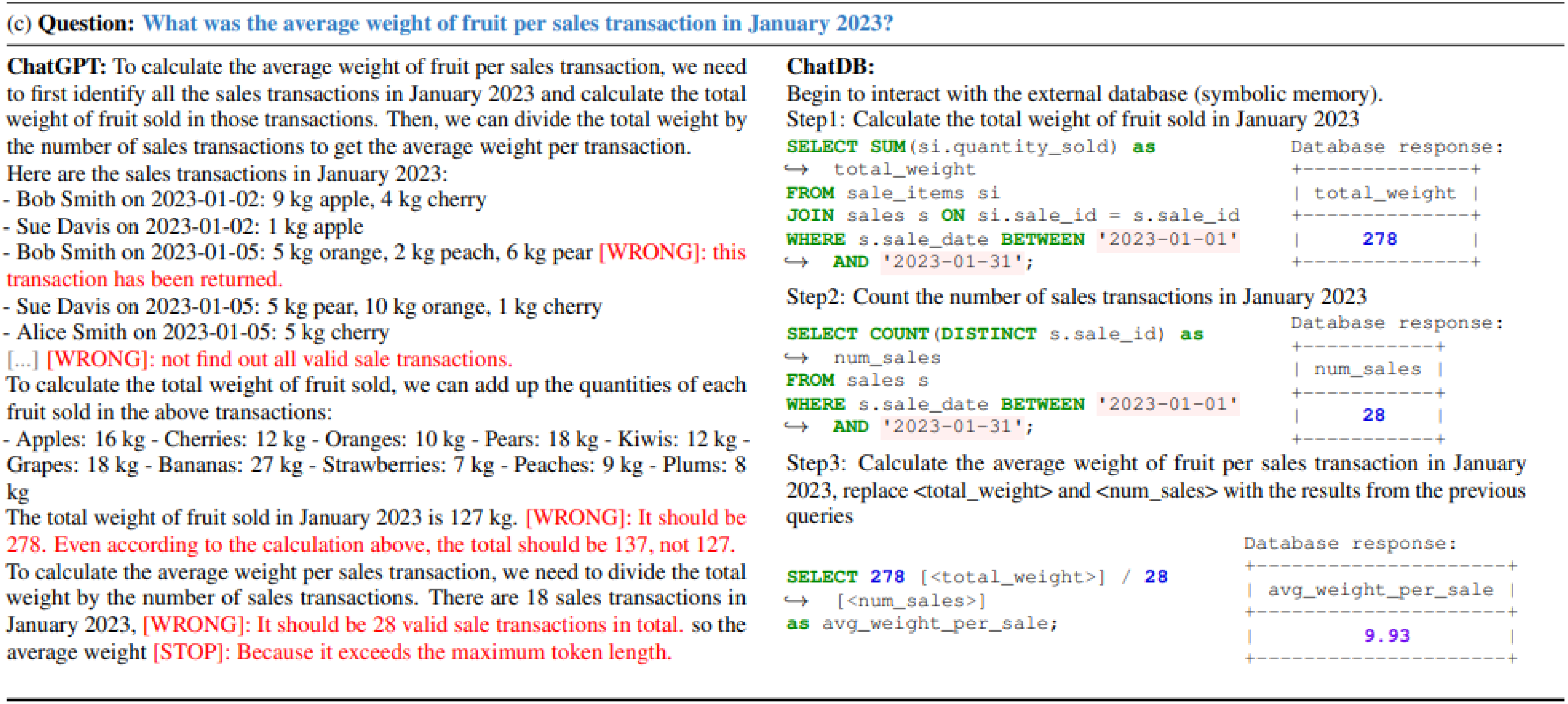
```
Database response:
+------------+---------+
| sale_date  | revenue |
+------------+---------+
| 2023-01-30 |  119.7  |
+------------+---------+
```

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Evaluation



Model	Easy	Hard	All	Accuracy
ChatGPT	10/15	1/35	11/50	22%
ChatDB (ours)	13/15	28/35	41/50	82%



(c) **Question:** What was the average weight of fruit per sales transaction in January 2023?

ChatGPT: To calculate the average weight of fruit per sales transaction, we need to first identify all the sales transactions in January 2023 and calculate the total weight of fruit sold in those transactions. Then, we can divide the total weight by the number of sales transactions to get the average weight per transaction.

Here are the sales transactions in January 2023:

- Bob Smith on 2023-01-02: 9 kg apple, 4 kg cherry
- Sue Davis on 2023-01-02: 1 kg apple
- Bob Smith on 2023-01-05: 5 kg orange, 2 kg peach, 6 kg pear [WRONG]: this transaction has been returned.
- Sue Davis on 2023-01-05: 5 kg pear, 10 kg orange, 1 kg cherry
- Alice Smith on 2023-01-05: 5 kg cherry

[...] [WRONG]: not find out all valid sale transactions.

To calculate the total weight of fruit sold, we can add up the quantities of each fruit sold in the above transactions:

- Apples: 16 kg - Cherries: 12 kg - Oranges: 10 kg - Pears: 18 kg - Kiwis: 12 kg - Grapes: 18 kg - Bananas: 27 kg - Strawberries: 7 kg - Peaches: 9 kg - Plums: 8 kg

The total weight of fruit sold in January 2023 is 127 kg. [WRONG]: It should be 278. Even according to the calculation above, the total should be 137, not 127.

To calculate the average weight per sales transaction, we need to divide the total weight by the number of sales transactions. There are 18 sales transactions in January 2023, [WRONG]: It should be 28 valid sale transactions in total. so the average weight [STOP]: Because it exceeds the maximum token length.

ChatDB:

Begin to interact with the external database (symbolic memory).

Step1: Calculate the total weight of fruit sold in January 2023

```
SELECT SUM(si.quantity_sold) as
↪  total_weight
FROM sale_items si
JOIN sales s ON si.sale_id = s.sale_id
WHERE s.sale_date BETWEEN '2023-01-01'
↪  AND '2023-01-31';
```

```
Database response:
+--------------+
| total_weight |
+--------------+
|      278     |
+--------------+
```

Step2: Count the number of sales transactions in January 2023

```
SELECT COUNT(DISTINCT s.sale_id) as
↪  num_sales
FROM sales s
WHERE s.sale_date BETWEEN '2023-01-01'
↪  AND '2023-01-31';
```

```
Database response:
+-----------+
| num_sales |
+-----------+
|     28    |
+-----------+
```

Step3: Calculate the average weight of fruit per sales transaction in January 2023, replace <total_weight> and <num_sales> with the results from the previous queries

```
SELECT 278 [<total_weight>] / 28
↪  [<num_sales>]
as avg_weight_per_sale;
```

```
Database response:
+---------------------+
| avg_weight_per_sale |
+---------------------+
|        9.93         |
+---------------------+
```

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

Limitations?

- 과일 가게 경영 시뮬레이션 이외에 다른 task에 적용하지 않아서 general task 에서의 성능을 보기 어려웠음

- DB에 날짜, 숫자 정보 이외에 더 복잡한 관계 정보를 표현할 수 있는지 알기 어려워 다른 task에 적용할 수 있는지 알기 어려움

- 그래도 DB를 symbolic memory로 활용하고, CoM을 통해 복잡한 task를 여러 단계의 SQL 쿼리로 분해하여 처리하는 과정이 신선했음…

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.

감사합니다
Q&A

Copyright © 2020 Natural Language Processing & Artificial Intelligence Lab.